



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/772,992

02/05/2004

James S. Miller

13768.493

5389

47973 7590 12/11/2008  
WORKMAN NYDEGGER/MICROSOFT  
1000 EAGLE GATE TOWER  
60 EAST SOUTH TEMPLE  
SALT LAKE CITY, UT 84111

EXAMINER

WANG, BEN C

ART UNIT

PAPER NUMBER

2192

MAIL DATE

DELIVERY MODE

12/11/2008

PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

<b>Office Action Summary</b>	<b>Application No.</b> 10/772,992	<b>Applicant(s)</b> MILLER ET AL.	
	<b>Examiner</b> BEN C. WANG	<b>Art Unit</b> 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

### Status

- 1) ☒ Responsive to communication(s) filed on 06 November 2008.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

### Disposition of Claims

- 4) ☒ Claim(s) 1, 4-23, 26 and 27 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1, 4-23, and 26-27 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

### Attachment(s)

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)          | 4) <input type="checkbox"/> Interview Summary (PTO-413)           |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____                                      |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)          | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| Paper No(s)/Mail Date _____  | 6) <input type="checkbox"/> Other: _____                          |

### **DETAILED ACTION**

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on November 6, 2008 has been entered.

2. Applicant's amendment dated November 6, 2008, responding to the Office action mailed July 10, 2008 provided in the rejection of claims 1, 4-23, and 26-27, wherein claims 1, 4, 20, and 22 have been amended.

Claims 1, 4-23, and 26-27 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Grier* and *Banks* - arts made of record, as applied hereto.

### ***Claim Rejections – 35 USC § 103(a)***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

Art Unit: 2192

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1, 4-6, 8-17, 20-22, and 26-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grier et al. (Pub. No. US 2002/0100017 A1) (hereinafter 'Grier' - art made of record) in view of Banks et al. (Pat. No. US 7,383,541 B1) (hereinafter 'Banks' - art made of record)

4. **As to claim 1** (Currently Amended), Grier discloses in a computerized system that includes one or more program computer-executable program components including one or more computer-executable requesting components configured to execute one or more computer-executable target components (e.g., Fig. 4, elements of "FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application's Request for Component with Version Independent Name; [0093] - ... the activation API receives the application request ...) in the computerized system, the target component comprising one of a non-overwritable library component and an overwritable platform component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...), a method of automatically providing

Art Unit: 2192

a computer-executable requesting component with access to an automatically determined version of a computer-executable target component upon request, comprising the acts of:

- upon receiving the one or more requests, identifying a versioning policy for each of the requested target components (e.g., Abstract - ... an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration ...);
- automatically determining from the identified versioning policy that the requested target component is a platform component or a library component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...); and

automatically providing the one or more requesting component with access to an appropriate version of the one or more target components on a differential basis from one target component to the next, wherein:

- if the requested target component is a library component, the requesting component is provided only a version of the target component that is specified by the requesting component (e.g., [0008] - ... allow an application to run with specified version of components bound thereto ...)

Further, Grier discloses an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration (e.g., Abstract), but does explicitly disclose other limitations stated below.

However, in an analogous art of *Method and Apparatus Providing Interoperation of Execution Image of Different Versions*, Banks discloses:

- receiving one or more requests from one or more requesting components for access by the one or more requesting components of one or more target components, wherein each request includes an indication of the lowest possible version of the target component that the requesting component can accept (e.g., Fig. 4, element 402A – Message Version 1); and
- if the requested target component is a platform component, the requesting component is automatically provided only the most recent servicing of the target component that is at least as recent as the lowest possible version of the target component specified by the requesting component (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation

Art Unit: 2192

infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Banks into the Grier's system to further provide other limitations stated above in the Grier system.

The motivation is that it would further enhance the Grier's system by taking, advancing and/or incorporating the Banks' system which offers significant advantages that a compatibility matrix specifies whether the versions are compatible, base-level compatible or incompatible; individual components of the execution images interoperate according to the results of individual session negotiations as once suggested by Banks (e.g., Abstract)

5. **As to claim 4** (Currently Amended) (incorporating the rejection in claim 1), Banks discloses the method, wherein the target component is a platform component, the method further comprising an act of identifying a more recent version of the target component in response to a request for an earlier version of the target even though the

Art Unit: 2192

more recent version and the earlier version are both accessible to the computerized system comprises identifying a more recent version of a platform component even though an earlier version of the platform component remained on the system when the more recent version was received at the computerized system (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

6. **As to claim 5** (Previously Presented) (incorporating the rejection in claim 1), Banks discloses the method wherein further comprising an act of identifying the



Art Unit: 2192

versioning policy of the specified lowest possible version of the target component when the specified lowest possible version of the target component is added to the computerized system (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

7. **As to claim 6** (Previously Presented) (incorporating the rejection in claim 1), Banks discloses the method wherein further comprising an act of storing, in the requesting component, version information that identifies the specified lowest possible version of the target component in the requesting component when the requesting

Art Unit: 2192

component is one or more of compiled, configured, installed, and run on the computerized system (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

8. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 1), Grier discloses the method wherein the request further includes a request for a specific version of the target component, wherein the requested specific version is different from the lowest possible version of the target component (e.g., [0008] - ... allow an application to run with specified version of components bound thereto ...)

9. **As to claim 9** (Previously Presented) (incorporating the rejection in claim 8), Grier discloses the method wherein the automatically determined appropriate version of the target component is different from the requested specific version of the target component that was requested (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...)

10. **As to claim 10** (Previously Presented) (incorporating the rejection in claim 1), Banks discloses further comprising receiving a plurality of new versions of the target component, wherein each of the new versions of the target component is associated with a different versioning policy (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-25)

11. **As to claim 11** (Previously Presented) (incorporating the rejection in claim 9), Banks discloses the method further comprising determining the appropriate version of the target component from among the specified lowest possible version of the target component and each of the plurality of new versions of the target component when the plurality of new versions of the target component (e.g., Table 1 – Example Compatibility

Art Unit: 2192

Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

12. **As to claim 12** (Original) (incorporating the rejection in claim 1), Banks discloses the method wherein the versioning policy is inserted into computer-executable instructions in the target component prior to one of installing, configuring, and executing the target component on the computerized system (e.g., Col. 10, Lines 52-60 – The compatibility matrix is stored in a way that it is accessible to the image of each peer ...)

Art Unit: 2192

13. **As to claim 13** (Previously Presented) (incorporating the rejection in claim 1), Banks discloses the method wherein the versioning policy is further identified in a plurality of version of the target component on the computerized system (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35)

14. **As to claim 14** (Previously Presented) (incorporating the rejection in claim 12), Grier discloses the method wherein each versioning policy in each version of the target component identifies a specific version of the requesting component configured to access that target component (e.g., [0008] - ... allow an application to run with specified version of components bound thereto ...)

15. **As to claim 15** (Original) (incorporating the rejection in claim 1), Banks discloses the method further comprising identifying a component scope that is associated with the target component (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35)

16. **As to claim 16** (Previously Presented) (incorporating the rejection in claim 1), Banks discloses the method wherein appropriate version of the target component is further automatically determined based on the identified component scope associated with the target component in addition to a determination of the lowest possible version that can be accepted (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A – V1 → V2 Transform; 404B – V2 → V3 Transform; 404C – V3 → V4 Transform; and 406A – Message Version

Art Unit: 2192

4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

17. **As to claim 17** (Previously Presented) (incorporating the rejection in claim 15), Banks discloses the method wherein the identified component scope specifies that access to the specified version of the target component is provided differently from the lowest possible version of the target component in one of a machine level, a process level, or a sub-process level (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A – V1 → V2 Transform; 404B – V2 → V3 Transform; 404C – V3 → V4 Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade

Art Unit: 2192

transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

18. **As to claim 20** (Currently Amended), Grier discloses in a computerized system that includes one or more computer-executable program components including one or more computer-executable requesting components that can request to access one or more computer-executable target components (e.g., Fig. 4, elements of “FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application’s Request for Component with Version Independent Name; [0093] - ... the activation API receives the application request ...) in the computerized system, the target component comprising one of a non-overwritable library component and an overwritable platform component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author

Art Unit: 2192

and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions.

The application author may also provide an application configuration that overrides the binding information in the application manifest ...), a method of automatically providing a computer-executable requesting component with access to an automatically determined version of a target component, comprising:

- a step for, upon receiving the request from the requesting component, automatically determining an appropriate version of the requested target component based on a versioning policy corresponding to the requested target component; and automatically allowing access to an appropriate version of the requested target component on a differential basis based on whether the requested target component is a platform component or a library component, such that the requesting component accesses the appropriate target component as it has been configured to do so, and such that the requesting component does not fail when requesting access to a component that has been upgraded (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...)



Further, Grier discloses an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration (e.g., Abstract), but does explicitly disclose other limitations stated below. However, in an analogous art of *Method and Apparatus Providing Interoperation of Execution Image of Different Versions*, Banks discloses:

- receiving one or more requests from one or more requesting components for access by the one or more requesting components of one or more target components, wherein each request includes an indication of the lowest possible version of the target component that the requesting component can accept (e.g., Fig. 4, element 402A – Message Version 1)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Banks into the Grier's system to further provide other limitations stated above in the Grier system.

The motivation is that it would further enhance the Grier's system by taking, advancing and/or incorporating the Banks' system which offers significant advantages that a compatibility matrix specifies whether the versions are compatible, base-level compatible or incompatible; individual components of the execution images interoperate according to the results of individual session negotiations as once suggested by Banks (e.g., Abstract)

Art Unit: 2192

19. **As to claim 21** (Previously Presented) (incorporating the rejection in claim 20),

Grier discloses the method wherein the step for allowing access to an appropriate version of the requested target component comprises the corresponding acts of:

- upon receiving the one or more requests, identifying a versioning policy for each of the requested target components (e.g., Abstract - ... an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration ...);
- automatically determining from the identified versioning policy that the requested target component is a platform component or a library component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...); and

automatically providing the one or more requesting components with access to an appropriate version of the one or more target components on a differential basis from one target component to the next, wherein:

- if the requested target component is a library component, the requesting component is provided only a version of the target component that is specified by the requesting component (e.g., [0008] - ... allow an application to run with specified version of components bound thereto ...)

Further, Banks discloses:

- if the requested target component is a platform component, the requesting component is automatically provided only the most recent servicing of the target component that is at least as recent as the lowest possible version of the target component specified by the requesting component (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target

component that is at least as recent as the lowest possible version) to the first peer 410 ...);

20. **As to claim 22** (Currently Amended), Grier discloses in a computerized system that includes one or more program components including one or more requesting components that can request to access one or more target components (e.g., Fig. 4, elements of “FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application’s Request for Component with Version Independent Name; [0093] - ... the activation API receives the application request ...) in the computerized system, the target component comprising one of a non-overwritable library component and an overwritable platform component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...), a method of automatically managing access of one or more versions of computer executable target component such that a computer-executable requesting component that accesses the target component continues to operate effectively after the target component has been upgraded with newer versions thereof, comprising the acts of:

- identifying that one or more requesting components are configured to execute a version of one or more computer-executable target components (e.g., Fig. 4,

Art Unit: 2192

elements of “FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application’s Request for Component with Version Independent Name; [0093] - ... the activation API receives the application request ...);

- automatically identifying a versioning policy for each of the one or more target components (e.g., Abstract - ... an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration ...);
- automatically determining for each of the one or more target components whether the target component is a platform component or a library component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...); and
- for each library component, determining based on the corresponding versioning policy to maintain all new versions of the target component, all the existing versions of the target component, and all of the previously installed version of the target component in the system at the same time (e.g., [0008] - ... allow an application to run with specified version of components bound thereto ...)

Further, Grier discloses an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration (e.g., Abstract), but does explicitly disclose other limitations stated below.

However, in an analogous art of *Method and Apparatus Providing Interoperation of Execution Image of Different Versions*, Banks discloses:

- for each platform component automatically determining based on the corresponding versioning policy for each platform component to remove any of the available versions of the platform component that are earlier than the version for which any of the one or more requesting components are configured (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message

406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...);

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Banks into the Grier's system to further provide other limitations stated above in the Grier system.

The motivation is that it would further enhance the Grier's system by taking, advancing and/or incorporating the Banks' system which offers significant advantages that a compatibility matrix specifies whether the versions are compatible, base-level compatible or incompatible; individual components of the execution images interoperate according to the results of individual session negotiations as once suggested by Banks (e.g., Abstract)

21. **As to claim 26** (Currently Amended), Grier discloses in a computerized system including one or more requesting components that are configured to access one or more target components (e.g., Fig. 4, elements of "FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application's Request for Component with Version Independent Name; [0093] - ... the activation API receives the application request ...) in the computerized system, the target component comprising one of a non-overwritable library component and an overwritable platform component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the

Art Unit: 2192

version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...), a computer storage product having computer-executable instructions stored thereon that, when executed, cause one or more processors in the computerized system to execute a method of automatically providing a computer-executable requesting component with access to an automatically determined version of a computer-executable target component upon request, comprising the acts of:

- upon receiving the one or more requests, identifying a versioning policy for each of the requested target components (e.g., Abstract - ... an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration ...);
- automatically determining from the identified versioning policy that the requested target component is a platform component or a library component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...); and



automatically providing the one or more requesting components with access to an appropriate version of the one or more target components on a differential basis for each target component, wherein:

- if the requested target component is a library component, the requesting component is provided only a version of the target component that is specified by the requesting component (e.g., [0008]  
- ... allow an application to run with specified version of components bound thereto ...)

Further, Grier discloses an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration (e.g., Abstract), but does explicitly disclose other limitations stated below. However, in an analogous art of *Method and Apparatus Providing Interoperation of Execution Image of Different Versions*, Banks discloses:

- receiving one or more requests from one or more requesting components for access by the one or more requesting components of one or more target components, wherein each request includes an indication of the lowest possible version of the target component that the requesting component can accept (e.g., Fig. 4, element 402A – Message Version 1);
- if the requested target component is a platform component, the requesting component is automatically provided only the most recent servicing of the target component that is at least as recent as the lowest possible version of the target component specified by the requesting component (e.g., Table 1 – Example

Art Unit: 2192

Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A –  $V1 \rightarrow V2$  Transform; 404B –  $V2 \rightarrow V3$  Transform; 404C –  $V3 \rightarrow V4$  Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Banks into the Grier's system to further provide other limitations stated above in the Grier system.

The motivation is that it would further enhance the Grier's system by taking, advancing and/or incorporating the Banks' system which offers significant advantages that a compatibility matrix specifies whether the versions are compatible, base-level

Art Unit: 2192

compatible or incompatible; individual components of the execution images interoperate according to the results of individual session negotiations as once suggested by Banks (e.g., Abstract)

22. **As to claim 27** (Currently Amended), Grier discloses in a computerized system including one or more requesting components that are configured to access one or more target components in the computerized system, the target component comprising one of a non-overwritable library component and an overwritable platform component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...), a computer program storage product having computer-executable instructions stored thereon that, when executed, cause one or more processors in the computerized system to execute a method of automatically managing access of one or more versions of computer-executable target component such that a computer-executable requesting component that accesses the computer-executable target component (e.g., Fig. 4, elements of "FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application's Request for Component with Version Independent Name; [0093] - ... the activation API receives the application

Art Unit: 2192

request ...) continues to operate effectively after the target component has been upgraded with newer versions thereof, comprising the acts of:

- identifying that one or more requesting components are configured to execute a version of one or more computer-executable target components (e.g., Fig. 4, elements of “FROM/TO SECOND PEER; 410 – FIRST PEER; Fig. 9, element 900 – Activation API Receives Application’s Request for Component with Version Independent Name; [0093] - ... the activation API receives the application request ...);
- automatically identifying a versioning policy for each of the one or more target components (e.g., Abstract - ... an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration ...);
- automatically determining for each of the one or more target components whether the target component is a platform component or a library component (e.g., [008] - ... allow an application to run with specified versions of components bound thereto, while allowing the application author and/or component publisher to change the version as desired ...; [0009] - ... the application provides an application manifest to specify any desired assembly versions. The application author may also provide an application configuration that overrides the binding information in the application manifest ...); and
- for each library component, determining based on the corresponding versioning policy to maintain all new versions of the target component, all the

Art Unit: 2192

existing versions of the target component, and all of the previously installed version of the target components in the system at the same time (e.g., [0008] - ... allow an application to run with specified version of components bound thereto ...)

Further, Grier discloses an application manifest specifies any desired assembly versions, which may be redirected to another version (overridden) by an application configuration (e.g., Abstract), but does explicitly disclose other limitations stated below.

However, in an analogous art of *Method and Apparatus Providing Interoperation of Execution Image of Different Versions*, Banks discloses:

- for each platform component automatically determining based on the corresponding versioning policy for each platform component to remove any of the available versions of the platform component that are earlier than the version for which any of the one or more requesting components are configured (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A – V1 → V2 Transform; 404B – V2 → V3 Transform; 404C – V3 → V4 Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation

Art Unit: 2192

infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...);

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Banks into the Grier's system to further provide other limitations stated above in the Grier system.

The motivation is that it would further enhance the Grier's system by taking, advancing and/or incorporating the Banks' system which offers significant advantages that a compatibility matrix specifies whether the versions are compatible, base-level compatible or incompatible; individual components of the execution images interoperate according to the results of individual session negotiations as once suggested by Banks (e.g., Abstract)

23. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Grier and Banks in view of Jonathan E. Cook (*Supporting Rapid Prototyping through Frequent and Reliable Deployment of Evolving Components*, 2001 IEEE, pp. 194-199) (hereinafter 'Cook' - art made previously applied)

24. **As to claim 7** (Previously Presented) (incorporating the rejection in claim 1),

Banks discloses the method further comprising:

- identifying one or more requesting components that are able to access a prior version of the target component (e.g., Fig. 4, elements 120 – Transformation Infrastructure; 404D V4→ V3 Transform; 404E V3→ V2 Transform; 404F V2 → V1 Transform; 402C Message Version 1 [DOWNGRADE]);
- identifying that none of the one or more requesting components are configured to request the prior version of the target component (e.g., Table 1 – Example Compatibility Matrix; Col. 10, Lines 24-35; Fig. 4, elements 120 – Transformation Infrastructure, 404A – V1 → V2 Transform; 404B – V2 → V3 Transform; 404C – V3 → V4 Transform; and 406A – Message Version 4 [UPGRADE]; Col. 13, Lines 8-15 – illustrating use of chained transformation function for message upgrading ...; Lines 33-46 - ... An upgrade transform upgrades the data from “version n” to “version n+1” ...; Col. 14, Lines 1-13 – ... the first peer 410 receives message 402A of v1 (i.e., the lowest possible version) from a second peer (i.e., requesting component). The first peer 410 (i.e., target component) calls transformation infrastructure 120 (i.e., the versioning policy), which applies a first transform 404A to result in converting the message to v2. The output of first transform 404A is chained or fed to the input of a second transform 404B that results in converting the v2 message to v3. The v3 message is chained to a third transform 404C that converts the v3 message to message 406A in v4. The transformation

Art Unit: 2192

infrastructure 120 returns message 406A of v4 (i.e., the most recent servicing of the target component that is at least as recent as the lowest possible version) to the first peer 410 ...);

Further, Grier and Banks do not explicitly disclose other limitations stated below.

However, in an analogous art of *Supporting Rapid Prototyping through Frequent and Reliable Deployment of Evolving Components*, Cook discloses:

- automatically deleting the prior version of the target component (e.g., P. 198, Left-Col., 2<sup>nd</sup> Par. – 3<sup>rd</sup> Par. - ... older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Cook into the Grier-Banks system to further provide other limitations stated above in the Grier-Banks system.

The motivation is that it would further enhance the Grier-Banks' system by taking, advancing and/or incorporating the Cook's system which offers significant advantages for decoupling the problem of version change and compatibility management from the deployment and testing of the new component version in the system as once suggested by Cook (e.g., Sec. 1 - Introduction, 2<sup>nd</sup> Para)

25. Claims 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grier in view of Banks and further in view of Steven Pratschner (*Simplifying Deployment and Solving DLL Hell with the .NET Framework*<sup>™</sup>, Nov. 2001, Microsoft Corporation<sup>™</sup>, pp. 1-12) (hereinafter 'Pratschner')



26. **As to claim 18** (Previously Presented) (incorporating the rejection in claim 1), Grier and Banks do not explicitly disclose the method further comprising identifying a servicing value associated with the requested target component.

However, in an analogous art of *Simplifying Deployment and Solving DLL Hell with the .NET Framework™*, Pratschner discloses the method wherein the requested target component is a library component, the method further comprising identifying a servicing value associated with the requested target component (e.g., P. 8, Sec. of “Custom Version Policy”, the vendor of a shared assembly may have shipped a service release to an existing assembly and would like all applications to being using the service release instead of the original version; these scenarios and others are supported in the .NET Framework™ through version policies; P. 11, Sec. of “Deployment” – the .NET Framework™ provides extensive code download services are integration with Windows Installer).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Pratschner into the Grier-Banks system to further provide the method wherein the requested target component is a library component, the method further comprising identifying a servicing value associated with the requested target component in Grier-Banks system.

The motivation is that it would further enhance the Grier-Banks system by taking, advancing and/or incorporating Pratschner’s system which offers significant advantages that the CLR records version information between pieces of an application and uses

Art Unit: 2192

that information at run time to ensure that the proper version of a dependency is loaded; version policies can be used by both application developers and administrators to provide some flexibility in choosing which version of a given assembly is loaded as once suggested by Pratschner (e.g., P. 12, Sec. of “Summary”)

27. **As to claim 19** (Previously Presented) (incorporating the rejection in claim 18), Pratschner discloses the method wherein identifying an appropriate version of the target component comprises identifying an updated servicing of a target component (e.g., P. 7, Sec. of “Version Policy” – including “Default Version Policy”, “Custom Version Policy”, and Version Policy Levels”; P. 8, Sec. of “Custom Version Policy”, the vendor of a shared assembly may have shipped a service release to an existing assembly and would like all applications to be using the service release instead of the original version; these scenarios and others are supported in the .NET Framework™ through version policies; P. 11, Sec. of “Deployment” – the .NET Framework™ provides extensive code download services are integration with Windows Installer)

28. Claim 23 is rejected under 35 U.S.C. 103(a) as being unpatentable over Grier in view of Banks and further in view of Eisenbach et al. (*Managing the Evolution of .NET™ Programs*, 2003, IFIP International Federation for Information Processing 2003 (hereinafter ‘Eisenbach’))

Art Unit: 2192

29. **As to claim 23** (Previously Presented) (incorporating the rejection in claim 22), Grier and Banks do not explicitly disclose the method wherein each target component includes a versioning value and a servicing value, the method further comprising:

- receiving an updated servicing of the existing version of one of the target component over a network from a network service provider;
- automatically overwriting the target component, wherein the existing version of the target component reflects the versioning value and a new servicing value;

However, in an analogous art of *Managing the Evolution of .NET™ Programs*, Eisenbach discloses the method wherein each target component includes a versioning value and a servicing value, the method further comprising:

- receiving an updated servicing of the existing version of one of the target component over a network from a network service provider (e.g., Fig. 4 – Distributed Dejavue.NET architecture, element of ‘Distributed Server’); and
- automatically overwriting the target component, wherein the existing version of the target component reflects the versioning value and a new servicing value (e.g., P. 186, 4<sup>th</sup> Par. – The design of the .NET assembly reveals some details about how this is accomplished, with each assembly carrying versioning information structured into four fields – Major, Minor (e.g., servicing value), Revision and Build. This allows many versions of the same DLL to co-exist in the system)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Eisenbach into the Grier-Banks

Art Unit: 2192

system to further provide the method wherein each target component includes a versioning value and a servicing value, the method further comprising:

- receiving an updated servicing of the existing version of one of the target component over a network from a network service provider;
- automatically overwriting the target component, wherein the existing version of the target component reflects the versioning value and a new servicing value.

The motivation is that it would further enhance the Grier-Banks' system by taking, advancing and/or incorporating the Eisenbach's system which offers significant advantages that a formal model was developed to assist in understanding the .NET mechanism and in describing our way of dealing with multiple versions and a tool was constructed to do so as once suggested by Eisenbach (e.g., Abstract, 2<sup>nd</sup> Par.)

### ***Conclusion***

30. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Art Unit: 2192

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/

Examiner, Art Unit 2192

/Tuan Q. Dam/

Supervisory Patent Examiner, Art Unit 2192